GPL162002 File System User Manual

V1.0 – Sep 22, 2006

Important Notice

GENERALPLUS TECHNOLOGY CO. reserves the right to change this documentation without prior notice. Information provided by GENERALPLUS TECHNOLOGY CO. is believed to be accurate and reliable. However, GENERALPLUS TECHNOLOGY CO. makes no warranty for any errors which may appear in this document. Contact GENERALPLUS TECHNOLOGY CO. to obtain the latest version of device specifications before placing your order. No responsibility is assumed by GENERALPLUS TECHNOLOGY CO. for any infringement of patent or other rights of third parties which may result from its use. In addition, GENERALPLUS products are not authorized for use as critical components in life support systems or aviation systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of GENERALPLUS.

GENERALPLUS Documentation

The following documents are available from GENERALPLUS. These documents provide useful information regarding the software programming and application designing.

GPL162002 File System User Manual Instruction set user's manual -This document explains the *GPL162002* file system instruction sets.

GPL162002 FILE SYSTEM **1**

USER MANUAL **1**

1 INTRODUCTION **6**

1.1 GENERAL DESCRIPTION 6

2 USER INTERFACE FUCTION **6**

2.1 OPEN 6
2.2 CLOSE 7
2.3 READ 8
2.4 WRITE 9
2.5 READB 9
2.6 WRITEB 10
2.7 LSEEK 11
2.8 MKDIR 12
2.9 RMDIR 13
2.10 CHDIR 13
2.11 GETCWD 14
2.12 UNLINK 15
2.13 RENAME 16
2.14 STAT 17
2.15 FSTAT 18
2.16 FS_INIT 19
2.17 FS_UNINIT 20
2.18 _GETFSERRCODE 20
2.19 _CLSFSERRCODE 21
2.20 _FINDFIRST 21
2.21 _FINDNEXT 23
2.22 _COPY 23
2.23 _FORMAT 24
2.24 _SETFATTR 25
2.25 _DELETEALL 26
2.26 _DEVICEMOUNT 27
2.27 _DEVICEUNMOUNT 28

2.28 _GETDISKFREE 29
2.29 VFSFREESPACE 30
2.30 FS_SAFEXIT 30
2.31 FS_REGISTERFD 31
2.32 CHANGECODEPAGE 31
2.33 CHECKFATTYPE 32
2.34 GETSECTORSPERCLUSTER 33
2.35 _GETCLUSTER 33
2.36 CLUS2PHY 34
2.37 GETDISKOFFILE 35
2.38 CREATFILEBYSIZE 36
2.39 UPDATADIR 37
2.40 FILEREPAIR 37
2.41 DELETEPARTFILE 38
2.42 INSERPARTFILE 39

3 FUNCTION REQUIREMENTS 40

3.1 USERGETDATE 40
3.2 USERGETTIME 40

4 LIMITS AND SUGGESTIONS 41

4.1 MAXIMUM OPEN FILE NUMBER 41
4.2 IMPLIED OPEN OPERATIONS 41
4.3 COPY OPERATION PERFORMANCE 42
4.4 MAXIMUM PATHNAME STRING LENGTH 42

Revision History

Revision	Date	By	Remark
v.1.0.0	2006-09-22	zhangzha	Modify for GPL162002 file system

Part I. Programming Development

1 Introduction

1.1 General Description

This guide describes the functionality and user API of DOS FAT/FAT32 File System for GPL162002 system.

2 User Interface Fuction

2.1 open

Open the specified file with the specified mode.

Function	Required Header
open	vfs.h

```
INT16 open(LPSTR path , INT16 open_flag)
```

Parameter

path

Pointer to a path name string.

flags

flags	Description
O_OPEN	Open a exists file
O_TRUNC	Open the file and truncate the file to zero length.
O_CREAT	If set, the file will be created if it doesn't already exist.
O_RDONLY	Open the file for read access.
O_WRONLY	Open the file for write access.
O_RDWR	Open the file for both reading and writing.
O_EXCL	If set, the open operation will fail when trying to create an existent file.

Return Values

The normal return value from open is a non-negative integer file descriptor. In the case of an error, a value of -1 is returned instead. **NOTE: Return value is a file node index. Maybe zero.**

Error Code List

attrib	Description
EACCES	The file exists but is not readable/writable as requested by the flags argument, the file does not exist and the directory is unwritable so it cannot be created.
EEXIST	Both O_CREAT and O_EXCL are set, and the named file already exists.
EISDIR	The flags argument specified write access, and the file is a directory.

ENFILE	The entire system, or perhaps the file system which contains the directory, cannot support any additional open files at the moment.
ENOENT	The named file does not exist, and O_CREAT is not specified.
ENAMETOOLONG	Filename specified too long.
EIO	For many devices, and for disk files, this error code indicates a hardware error.
ENAMEINVALID	Invalid character detected in the filename string.
ENOSPC	No free directory entry left in root directory on a FAT16 or FAT12 file system or the file system doesn't have enough room to extend the directory.

Example

```
#include  <stdio.h>
#include  <string.h>
#include  "vfs.h"
void main (void) {
        int fp, err;
        char string[] = "Hello , world\n";
        // open with create options
        if ((fp = open ((LPSTR)"A:\\test.txt", O_CREAT | O_RDWR)) == -1)
                printf ("The file 'test.txt' was not created\n");
        else
                printf ("The file 'test.txt' was created\n");
        // write string to file
        write (fp, (UINT32)string<<1, strlen(string));
        close (fp);
}
```

2.2 close

The function close the file specified by the file node index and flush the buffers associates to the file.

Function	Required Header
close	vfs.h

`int close (int filedes)`

Parameter

filedes

File node index, maybe it's "open" 's return value.

Return values

The normal return value from close is 0; a value of -1 is returned in case of failure.

Error Code List

attrib	Description

EBADF	The *filedes* argument is not a valid file descriptor.

Example

```
fp = open ((LPSTR)".\\Hello.txt", O_CREAT | O_RDWR);
write (fp, (UINT32)buffer<<1, len);
close (fp);  // system will write the data to disk
```

2.3 read

The read function reads up to size bytes from the file with descriptor filedes, storing the results in the buffer.

Function	Required Header
read	vfs.h

`int read (int filedes, unsigned long buffer, unsigned int size)`

Parameter

filedes

File node index, maybe it's "open" 's return value.

buffer

Buffer pointer. It's data pointer given to specify an offset in the SRAM. Note it must be byte address.

size

Reads up to *size* bytes from the file. Note it must be byte size.

Return values

The return value is the number of bytes actually read. This might be less than *size*; In case of an error, read returns -1.

Error codes list

Value	Description
EBADF	The filedes argument is not a valid file descriptor, or is not open for reading.
EIO	For many devices, and for disk files, this error code indicates a hardware error.

Example

```
int fp;
fp = open ((LPSTR)".\\Hello.txt", O_CREAT | O_RDWR);
write (fp, (UINT32)buffer<<1, 14); // write 14 bytes start form buffer to file
close (fp);  // system will write the data to disk
fp = open((LPSTR)".\\Hello.txt", O_RDONLY);
read (fp, (UINT32)buffer<<1, 14);   // read 14 byte form the file into buffer
close (fp);
```

2.4 write

The write function writes up to size bytes from buffer to the file with descriptor filedes.

Function	Required Header
write	vfs.h

```
int write (int filedes, unsigned long buffer, unsigned int size)
```

Parameter

filedes

File node index, maybe it's "open" 's return value

buffer

Buffer pointer. It's data pointer given to specify an offset in the SRAM. Note it must be byte address.

size

Writes up to *size* bytes from buffer to the file. Note it must be byte size.

Return values

The return value is the number of bytes actually written. This may be *size*, but can always be smaller. Your program should always call write in a loop, iterating until all the data is written. In the case of an error, write returns -1.

Error code list

Value	Description
EBADF	The *filedes* argument is not a valid file descriptor, or is not open for writing.
EIO	For many devices, and for disk files, this error code indicates a hardware error.
EACCES	File access mode error. Write to a file which is opened with O_RDONLY access mode.
ENOSPC	Media is full. File system cannot allocate any free cluster for the new write operation.

Example

```
// creat a new file
fp = open ((LPSTR)".\\Hello.txt", O_CREAT | O_RDWR);
// write a string data to file
write (fp, 0, 14);    // Write the first 14 bytes in SRAM into the file
close (fp);
```

2.5 readB

The readB function reads up to size bytes from the file with descriptor filedes, storing the results in the buffer.

Function	Required Header
readB	vfs.h

```
int readB (int filedes, unsigned long buffer, unsigned int size)
```

Parameter

filedes

File node index, maybe it's "open" 's return value.

buffer

Buffer pointer. It's data pointer given to specify an offset in the SRAM. Note it must be word address.

size

Reads up to *size* bytes from the file. Note it must be byte size.

Return values

The return value is the number of bytes actually read. This might be less than *size*; In case of an error, read returns -1.

Error codes list

Value	Description
EBADF	The filedes argument is not a valid file descriptor, or is not open for reading.
EIO	For many devices, and for disk files, this error code indicates a hardware error.

Example

```
int fp;
fp = open((LPSTR)".\\Hello.txt", O_RDONLY);
readB (fp, (UINT32)buffer, 14);// read 14 byte form the file into the buffer
close (fp);
```

2.6 writeB

The writeB function writes up to size bytes from buffer to the file with descriptor filedes.

Function	Required Header
writeB	vfs.h

`int writeB (int filedes, unsigned long buffer, unsigned int size)`

Parameter

filedes

File node index, maybe it's "open" 's return value

buffer

Buffer pointer. It's data pointer given to specify an offset in the SRAM. Note it must be word address.

size

Writes up to *size* bytes from buffer to the file. Note it must be byte size.

Return values

The return value is the number of bytes actually written. This may be *size*, but can always be smaller. Your program should always call write in a loop, iterating until all the data is written. In the case of an error, write returns -1.

Error code list

Value	Description
EBADF	The *filedes* argument is not a valid file descriptor, or is not open for writing.
EIO	For many devices, and for disk files, this error code indicates a hardware error.
EACCES	File access mode error. Write to a file which is opened with O_RDONLY access mode.
ENOSPC	Media is full. File system cannot allocate any free cluster for the new write operation.

Example

```
// creat a new file
fp = open ((LPSTR)".\\Hello.txt", O_CREAT | O_RDWR);
// write a string data to file
writeB (fp, buffer, 14);   // Write the first 14 bytes in SRAM into the file
close (fp);
```

2.7 lseek

The lseek sets the read-write file pointer for the open file specified by the fd.

Function	Required Header
lseek	vfs.h

`long lseek (int fd, long offset, int whence)`

Parameter

fd

Index of the file node.

offset

The offset will be set.

Whence

Value	Description
SEEK_SET	Sets the file pointer to the value of the *offset* parameter.
SEEK_CUR	Sets the file pointer to its current location plus the value of the *offset* parameter.
SEEK_END	Sets the file pointer to the size of the file plus the value of the *offset* parameter.

Return values

The return value from lseek is normally the resulting file position, measured in bytes from the beginning of the file. You can use this feature together with SEEK_CUR to read the current file position. If the file position cannot be changed, or the operation is in some way invalid, lseek returns a value of -1.

Error code list

Value	Description
EBADF	The *fd* parameter is not an open file descriptor.
EINVAL	The *whence* argument value is not valid or seek to an invalid position.

NOTE

When a file was “lseeked” to a place where is out of the size of the file and a write operation followed, file size extending operation will be performed. It means that file system will read and adapt the FAT for free cluster allocation. The clusters are dirty because file system never try to clean the rubbish data on the free clusters.

Example

```
int fid;
// open a text file for reading
fid = open ((LPSTR)“a:\\tempfile.txt”, D_RDONLY);
// set the read-write file pointer
lseek (fid, 100, SEEK_SET);
```

2.8 mkdir

Directories are created with the mkdir function.

Function	Required Header
mkdir	vfs.h

`int mkdir (LPSTR filename)`

Parameter

filename

Pointer to a string specify the directory name will be created.

Return values

A return value of 0 indicates successful completion, and -1 indicates failure.

Error code list

Value	Description
EACCES	Write permission is denied for the parent directory in which the new directory is to be added.
EEXIST	A file named filename already exists.
ENOSPC	The file system doesn't have enough room to create the new directory.
ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist.
EIO	For many devices, and for disk files, this error code indicates a hardware error.
ENAMETOOLONG	Filename specified too long.
ENAMEINVALID	Invalid character detected in the filename string.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
void create_temp_directory ()  {
    if (mkdir ((LPSTR)“a:\\temp” ) == -1)   {
        printf (“can not create temporary directory\n”);
    }
    else {
        printf (“temporary directory created\n”);
    }
}
```

2.9 rmdir

The rmdir function deletes a directory.

Function	Required Header
rmdir	vfs.h

`int rmdir (LPSTR filename)`

Parameter

filename

Pointer to a string specify the directory name will be deleted.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
ENOTEMPTY	The directory to be deleted is not empty.
EACCES	Write permission is denied for the directory from which the file is to be removed.
ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.
EIO	For many devices, and for disk files, this error code indicates a hardware error.

Example

```
/* delect a temporary directory */
if (rmdir ((LPSTR)"a:\\temp") == -1)
    printf ("rmdir failed\n");
```

2.10 chdir

This function is used to set the process's working directory to *filename*.

Function	Required Header
chdir	vfs.h

`int chdir (LPSTR filename)`

Parameter

filename

Pointer to directory string.

Return values

The normal, successful return value from chdir is 0. A value of -1 is returned to indicate an error.

Error code list

Value	Description
ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist,
ENOTDIR	A file that is referenced as a directory component in the file name exists, but it isn't a directory.
EIO	For many devices, and for disk files, this error code indicates a hardware error.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
// change current directory
if (chdir ((LPSTR)“a:\\temp”) == -1) { // return value -1 means error occurred
     printf (“change current directory failed\n”);}
else {
     printf (“change current directory successful\n”);
}
```

2.11 getcwd

Get current directory if success return buffer address else return NULL.

Function	Required Header
getcwd	vfs.h

`LPSTR getcwd (LPSTR buffer, int size)`

Parameter

buffer

Pointer to directory string buffer.

size

Maximum length of the directory string can be stored.

Return values

The return value is buffer on success and a null pointer on failure.

Error code list

Value	Description
EINVAL	The *size* argument is zero and *buffer* is not a null pointer.

ERANGE	The *size* argument is less than the length of the working directory name. You need to prepare a bigger array and try again.

Example

```
…
int size = 100;
char buffer[100];
if (getcwd (buffer, size) == buffer)    {
     printf (“\nCurrent path: %s”, buffer);
}
```

2.12 unlink

Delete the specified file.

Function	Required Header
unlink	vfs.h

`int unlink (LPSTR filename)`

Parameter

filename

The file will be unlinked.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
EACCES	Write permission is denied for the directory from which the file is to be removed or the file is busy.
ENOENT	The *filename* to be deleted doesn't exist.
EISDIR	Unlink cannot be used to delete the name of a directory. To avoid such problems, use rmdir to delete directories.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
if (unlink ((LPSTR)“a:\\tempfile.txt”) == -1)    {
     switch (_getfserrcode ())  {
     case EACCES:
          …
     }
}
else {
     printf (“file has been deleted\n”);
}
```

2.13 rename

The function can be used to move or rename a file or a directory.

Function	Required Header
rename	vfs.h

`int rename (LPSTR oldname, LPSTR newname)`

Parameter

oldname

The old name of the file.

newname

The new name of the file.

Return values

-1 returned when error occurred.

Error code list

Value	Description
EACCES	One of the directories containing *newname* or *oldname* refuses write permission; or *newname* and *oldname* are directories and write permission is refused for one of them.
EEXIST	The file or directory *newname* is already existed.
ENOENT	The file *oldname* doesn't exist.
ENOSPC	The directory that would contain *newname* has no room for another entry, and there is no space left in the file system to expand it.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
// change the filename of “file1” into “file2”
int res;
res = rename ((LPSTR)“a:\\file1”, (LPSTR)“a:\\file2”);
if (res == -1)
    printf (“rename failed\n”);
```

2.14 stat

The stat function fills the specified structure with the information about the specified file.

Function	Required Header
stat	vfs.h

```
int stat (LPSTR filename, struct stat *buf)
```

Parameter

filename

Pointer to a pathname string.

buf

Pointer to a stat structure.

Data Structure

```
struct stat  {
    unsigned char st_mode;// access attribute of the file, see file mode bitwise mask
    unsigned long  st_size;// the size of the normal file in byte
    unsigned long st_mtime;       // the last modification time
};
```

File Mode Bitwise Mask

Value	Description
S_READ_ONLY	Read only attribute.
S_HIDDEN	Hide attribute. A hidden file has this attribute.
S_SYSTEM	System file attribute.
S_DIRECTORY	Directories have this attribute.
S_ARCHIVE	Normal file attribute.

Return values

The return value is 0 if the operation is successful, or -1 on failure.

Error code list

Value	Description
EINVAL	Parameter list error: neither the *filename* nor the *buf* can be NULL.
ENOENT	The file named by filename doesn't exist.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
…
/* Compare two files' last modification times */
struct stat statbuf;
time_t time1;
int res;
res = stat ((LPSTR)"file1.txt", &statbuf);
if (res)
    return -1;
time1 = statbuf.st_mtime;
res = stat ("file2.txt", &statbuf);
if (res)
    return -1;
if (time1 > statbuf.st_mtime)
    printf ("file1.txt is more recent");
else
    printf ("file2.txt is more recent");
…
```

2.15 fstat

The fstat function fills the specified structure with the information about the specified file.

Function	Required Header
fstat	vfs.h

```
int fstat(int handle, struct stat *statbuf)
```

Parameter

`handle`

Index of the file node.

buf

Pointer to a stat structure.

Data Structure

struct stat {

unsigned char st_mode;// access attribute of the file, see **file mode bitwise mask**

unsigned long st_size;// the size of the normal file in byte

unsigned long st_mtime;// the last modification time

};

File Mode Bitwise Mask

Value	Description
S_READ_ONLY	Read only attribute.
S_HIDDEN	Hide attribute. A hidden file has this attribute.
S_SYSTEM	System file attribute.
S_DIRECTORY	Directories have this attribute.
S_ARCHIVE	Normal file attribute.

Return values

The return value is 0 if the operation is successful, or -1 on failure.

Error code list

Value	Description
EINVAL	Parameter list error: neither the *filename* nor the *buf* can be NULL.
ENOENT	The file named by filename doesn't exist.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
…
struct stat statbuf;
int res;
int fd;
fd = open((LPSTR)“Hello.txt”, O_RDONLY);
res = fstat (fd, &statbuf);
if (res)
    return -1;
close(fd);
…
```

2.16 fs_init

Initialize file system.

Function	Required Header
fs_init	vfs.h

```
void fs_init (void)
```

Parameter

No parameter.

Return values

No return value.

Note:

All the global variables of file system will be forced into the initial values. Typically this function should be

used only once at the initialization part of your program.

Example

```
fs_init ();
_devicemount (0);
fp = open("a:\\Hello.txt" , O_CREAT | O_RDWR);
write (fp, 0, 14);
close(fp);
_deviceunmount (0);
```

2.17 fs_uninit

unmount all the device.

Function	Required Header
fs_uninit	vfs.h

```
void fs_uninit (void)
```

Parameter

No parameter.

Return values

No return value.

Example

```
fs_init ();     //initial all the global variable, and mount all the device.

fs_uninit ();   //unmount all the device.
```

2.18 _getfserrcode

Get the last error code of the file system.

Function	Required Header
_getfserrcode	vfs.h

```
int _getfserrcode ()
```

Parameter

No parameter required.

Return values

Return the last error code value. It can be one of the following values.

Value	Description
ENOENT	No such file or directory
EINVACC	Invalid access mode
EBADF	Bad file number
EINVFNC	Invalid function number
ENOMEM	Not enough core
ERANGE	Not enough core
EACCES	Permission denied
EEXIST	File exists
EISDIR	Target specified not a file but a directory
EINVAL	Invalid argument
EMFILE	Too many open files
ENOSPC	No space left on device
ENOTEMPTY	Directory is not empty
EIO	I/O operation error
ENOTDIR	Not directory
ENFILE	File not found
EROFS	Incorrect access mode
EPERM	Target is a directory
EBUSY	Target device is busy
ENAMETOOLONG	Specified path name or file name too long.
ENAMEINVALID	Invalid character detected in the filename string.

2.19 _clsfserrcode

Set the global error code value to zero.

Function	Required Header
_clsfserrcode	vfs.h

```
void _clsfserrcode ()
```

Parameter

No parameter required.

Return values

No reture value.

2.20 _findfirst

Find the first appointed name and attribute's file.

Function	Required Header
_findfirst	Vfs.h

```
int _findfirst (LPSTR pathname, struct f_info * ffblk, int attrib)
```

Parameter

pathname

Pointer to a pathname string.

ffblk

Pointer to a f_info structure.

```
struct f_info
{
    unsigned char f_attrib;   /* file attribute */
    unsigned int f_time;      /* file time */
    unsigned int f_date;      /* file date */
    unsigned long f_fsize;    /* file size */
    char   f_name[256];       /* file name */
};
```

attrib

attrib	Description
D_RDONLY	Read-only file attribute
D_HIDDEN	Hidden file attribute
D_SYSTEM	System file attribute
D_DIR	Directory attribute
D_ARCHIVE	Archive file attribute
D_ALL	All above attribute

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist, or when a component is a symbolic link whose target file does not exist.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.
EIO	For many devices, and for disk files, this error code indicates a hardware error.

Example

```
…
void list_file (char * pattern){
    struct f_info finfo;
    int idx = 0;

    printf ("\nList \"%s\":", pattern);
```

2.21 _findnext

Find next appointed name and attribute's file.

Function	Required Header
_findnext	vfs.h

```
int _findnext (struct f_info *finfo)
```

Parameter

finfo

File information struct.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
ENOENT	No more file fit the specified file name pattern and the file attribute condition.
EIO	For many devices, and for disk files, this error code indicates a hardware error.

Example

See _findfirst () example.

2.22 _copy

The function can be used to make a copy for a file.

Function	Required Header
_copy	vfs.h

`int _copy (LPSTR srcfile, LPSTR destfile)`

Parameter

srcfile

It is a source path of file and file name.

destfile

It is a destination path of file and file name.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist. Or the file named by filename doesn't exist.
EEXIST	A file named filename already exists.
EMLINK	The parent directory has too many links (entries).
ENOSPC	The file system doesn't have enough room to create the new directory.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
int backup_a_c (void) {
     int res;
     res = _copy ((LPSTR)“a:\\a.c”, (LPSTR)“a:\\backup\\a.c”);
     if (!res)    {
          return 0;
     }
     else {
          printf (“file backup error”);
          return –1;
     }
}
```

2.23 _format

Create a file system with the specified driver.

Function	Required Header
_format	vfs.h

`int _format (unsigned char drv, unsigned char fstype)`

Parameter

drv

Zero-based driver set index. For example, index for SD card is 0.

fstype

Specify the file system. Should be one of these values.

Value	Description
FAT16_Type	Format the disk with FAT16 storage format.
FAT32_Type	Format the disk with FAT32 storage format.
FORCE_FAT16_Type	Format the disk with FAT16 storage format and force rewrite MBR
FORCE_FAT32_Type	Format the disk with FAT32 storage format and force rewrite MBR.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
EIO	For many devices, and for disk files, this error code indicates a hardware error.
EINVAL	A argument list error detected. May be the *drv* value is larger than NBLKDEV or the *fstype* is greater than FAT32_Type.
EBUSY	The target device is a mounted device. Can not format a mounted device.

Note:

Format should be used on a device which is not mounted.

Example

```
fs_init ();
if (_format (0, FAT16_Type))   /* make a FAT16 file system on the SD card */
    while(1);
/* mount for other operations after _format() */
_devicemount (0);
```

2.24 _setfattr

Change the attributes of the specified file.

Function	Required Header
_setfattr	vfs.h

`int _setfattr (LPSTR filename, unsigned short attr)`

Parameter

filename

Pointer to a path name string which specify the file name whose attributes will be modified.

attr

The attributes will be applied on the specified file.

Parameter Values

Value	Description
D_NORMAL	Normal file without any special attribute.
D_RDONLY	Read only attribute.
D_HIDDEN	Hidden file attribute.
D_SYSTEM	System file attribute.
D_ARCHIVE	Archive file attribute.

The value of attr can be the bitwise OR result of the upper bit values.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
ENOENT	The file doesn't exist.
EINVAL	Parameter list error: the filename can not be NULL and the attribute must be the bitwise or result of some of S_READ_ONLY, S_HIDDEN, S_SYSTEM, S_ARCHIVE bit values.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.
EIO	For many devices, and for disk files, this error code indicates a hardware error.

Example

```
unsigned short attr = D_RDONLY | D_HIDDEN;
_setfattr ((LPSTR)“a:\\tempfile.txt”, attr);
```

2.25 _deleteall

Delete all files and folders in the specified directory.

Function	Required Header
_deleteall	vfs.h

`int _deleteall (LPSTR dir)`

Parameter

dir

To appoint path.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description

ENOENT	This error is reported when a file referenced as a directory component in the file name doesn't exist. Or the file named by filename doesn't exist.
EACCES	Write permission is denied for the directory from which the file is to be removed. The deleteall process will terminated after such an accident.
EIO	Low level I/O error.
ENFILE	The entire file system cannot allocate any file node structure variable for search at the moment. See “Limits and Suggestions”.

Example

```
/* _deleteall can be used to empty a directory */
/* Example, empty a temporary directory */
int res;
res = _deleteall ((LPSTR)“a:\\temp”);
if (res) {
    …
}
```

2.26 _devicemount

Mount a disk, load the information about the device and the file system information on the device.

Function	Required Header
_devicemount	vfs.h

`int _devicemount (unsigned char diskid)`

Parameter

diskid

To appoint mounted disked.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
EINVAL	Parameter list value error.
EBUSY	The device specified by *diskid* is busy.
EIO	Low level I/O error.

Example

```
fs_init ();
if (_devicemount (0))
    while(1);
```

2.27 _deviceunmount

Umount the specified device, flush all cached data associates to the device.

Function	Required Header
_deviceunmount	vfs.h

```
int _deviceunmount (char diskID)
```

Parameter

diskID

To appoint mounted disked.

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
EINVAL	Parameter list value error.
EBUSY	The device specified by *diskid* is busy.

Example

```
fs_init ();
_devicemount (0);
_deviceunmount(0);
```

2.28 _getdiskfree

Get information about the space of the specified device.

Function	Required Header
_getdiskfree	vfs.h

```
int _getdiskfree (unsigned char driver, struct _diskfree_t * dfreep)
```

Parameter

driver

Specify the device zero based index.

dfreep

The struct of device information

struct _diskfree_t

{

unsigned long total_clusters;

unsigned long avail_clusters;

unsigned sectors_per_cluster;

unsigned bytes_per_sector;

};

Return values

This function returns 0 on successful completion, and -1 on error.

Error code list

Value	Description
EINVAL	Parameter list value error.

Example

```
…
struct _diskfree_t space_info
_ getdiskfree (0, &space_info);    // get space informations about device a
printf (“Free clusters: %d”, space_info.avail_clusters);
…
```

2.29 vfsFreeSpace

Get the free space of device.

Function	Required Header
vfsFreeSpace	vfs.h

```
long vfsFreeSpace(short driver)
```

Parameter

driver

Specify the device zero based index.

Return values

The free space of device.Count in byte.

Error code list

Value	Description
EINVAL	Parameter list value error.

Example

```
…
long freespace;
freespace  = _vfsFreeSpace (0); // get free space of device
printf (“Free space: %d”, freespace);
…
```

2.30 fs_safexit

Close all the file that opened and unregiste.

Function	Required Header
fs_safexit	vfs.h

```
void fs_safexit(void)
```

Parameter

No parameter.

Return values

No return value.

Example

```
…
fs_safexit();
…
```

2.31 fs_registerfd

Registe the file handle that opened so when call the function . fs_safexit() it will not be closed.

Function	Required Header
fs_registerfd	vfs.h

```
void fs_registerfd(int fd)
```

Parameter

fd

Index of the file node.

Return values

No return value.

Example

```
…
fd = open((LPSTR)”test.txt”, O_CREAT|O_RDWR);
if(fd < 0)
    return;
fs_registerfd (fd);


fs_safexit();     //the file fd will not be closed, and other file that unregiste
                   //will be force closed
```

2.32 ChangeCodePage

Change the Unicode page.

Function	Required Header
ChangeCodePage	vfs.h

```
int ChangeCodePage(unsigned int wCodePage);
```

Parameter

wCodePage

The unicode page.

Specify the file system. Should be one of these values.

Value	Description
UNI_GBK	Select unicode page is GBK.
UNI_BIG5	Select unicode page is BIG5.
UNI_SJIS	Select unicode page is SJIS.
UNI_ENGLISH	Select unicode page is ENGLISH.
UNI_ARABIC	Select unicode page is ARABIC.
UNI_INVALID_TAB	Valid unicode page, it will default change to UNI_ENGLISH.

Return values

The unicode page.

NOTE

To support UNI_GBK, UNI_BIG5, UNI_SJIS, UNI_ARABIC, it must be load corresponding Unicode lookup table into NAND FLASH reserve area or code area.

Example

```
…
ChangeCodePage(UNI_BIG5); //change Unicode page to BIG5


…
```

2.33 checkfattype

get the fat type of the device.

Function	Required Header
checkfattype	vfs.h

```
int checkfattype(int drv);
```

Parameter

drv

Zero-based driver set index. For example, index for SD card is 0.

Return values

Fat type.

Value	Description
FAT16_Type	Format the disk with FAT16 storage format.
FAT32_Type	Format the disk with FAT32 storage format.

Example

```
…
int fattype;
fattype  = checkfattype(0);
…
```

2.34 GetSectorsPerCluster

get the sector number of one cluster.

Function	Required Header
GetSectorsPerCluster	vfs.h

```
int GetSectorsPerCluster (int drv);
```

Parameter

drv

Zero-based driver set index. For example, index for SD card is 0.

Return values

Sector number.

Example

```
…
int n;
n  = GetSectorsPerCluster (0); // the return value n is sector number of one cluster
…
```

2.35 _GetCluster

Get the cluster address of the file now seek to.

Function	Required Header
_GetCluster	vfs.h

```
long _GetCluster (int fd)
```

Parameter

fd

Index of the file node.

Return values

The cluster of the file now seek to.

Example

```
…
int i;
long buf[100];
fd = open((LPSTR)”test.txt”, O_RDWR);
for(i = 0; i < 100; i++)
{
     lseek(fd, i*0x200, 0);
     buf[i] = _ GetCluster(fd);
}
close (fd);

…
```

2.36 Clus2Phy

Transfer the cluster address to sector address.

Function	Required Header
Clus2Phy	vfs.h

```
long Clus2Phy (int dsk, unsigned long cl_no)
```

Parameter

dsk

Zero-based driver set index. For example, index for SD card is 0.

cl_no

cluster address.

Return values

The sector address.

Example

```
…
int i;
long buf[100];
long cluster;
fd = open((LPSTR)”test.txt”, O_RDWR);
for(i = 0; i < 100; i++)
{
     lseek(fd, i*0x200, 0);
     cluster = _GetCluster(fd);
     buf[i] = Clus2Phy(0, cluster);          //the sector address
}
close (fd);
…
```

2.37 GetDiskOfFile

Get the file in witch disk.

Function	Required Header
GetDiskOfFile	vfs.h

```
int GetDiskOfFile(int fd)
```

Parameter

fd

Index of the file node.

Return values

The disk no.

Example

```
…
int dsk;
fd = open((LPSTR)”test.txt”, O_RDWR);
dsk = GetDiskOfFile(fd); //get the file in witch disk.
close (fd);
…
```

2.38 CreatFileBySize

Creat a file by sppointed size.

Function	Required Header
CreatFileBySize	vfs.h

```
int CreatFileBySize (LPSTR path ,unsigned long size)
```

Parameter

path

The file will be creat.

Size

The size of the file that would be created.

Return values

The normal return value from open is a non-negative integer file descriptor. In the case of an error, a value of -1 is returned instead. **NOTE: Return value is a file node index. Maybe zero.**

NOTE

If creat file success, the file data is unpredictable.

Error Code List

attrib	Description
EACCES	The file exists but is not readable/writable as requested by the flags argument, the file does not exist and the directory is unwritable so it cannot be created.
EEXIST	Both O_CREAT and O_EXCL are set, and the named file already exists.
EISDIR	The flags argument specified write access, and the file is a directory.
ENFILE	The entire system, or perhaps the file system which contains the directory, cannot support any additional open files at the moment.
ENOENT	The named file does not exist, and O_CREAT is not specified.
ENAMETOOLONG	Filename specified too long.
EIO	For many devices, and for disk files, this error code indicates a hardware error.
ENAMEINVALID	Invalid character detected in the filename string.
ENOSPC	No free directory entry left in root directory on a FAT16 or FAT12 file system or the file system doesn't have enough room to extend the directory.

Example

```
fd = CreatFileBySize((LPSTR)" HELLO.TXT" ,10000);
if(fd < 0)
    return – 1;


close(fd);
```

2.39 UpdataDir

Write the dir info back to the device and flush all the buffers but not close file.

Function	Required Header
UpdataDir	vfs.h

```
int UpdataDir(int fd)
```

Parameter

fd

Index of the file node.

Return values

A return value of 0 indicates successful completion, and -1 indicates failure.

Example

```
…
int dsk;
fd = open((LPSTR)”test.txt”, O_RDWR);
for(;;)
{
     write(fd, (UINT32)buffer<<1, 1024);
     UpdataDir(fd);    //write back the dir info, and all the fs buffer
}
close (fd);
```

2.40 FileRepair

Repair the bad file that may cause by power supply broken.

Function	Required Header
FileRepair	vfs.h

```
int FileRepair(int fd)
```

Parameter

fd

Index of the file node.

Return values

A return value of 0 indicates successful completion, and -1 indicates failure.

Example

```
…
int dsk;
fd = open((LPSTR)”test.txt”, O_RDWR);
FileRepair(fd);
close (fd);
…
```

2.41 DeletePartFile

Delete some data of the the file.

Function	Required Header
DeletePartFile	vfs.h

```
int DeletePartFile(int fd, unsigned long offset, unsigned long length)
```

Parameter

fd

Index of the file node.

offset

The address start to delete.

Length

The length want to delete.

Return values

A return value of 0 indicates successful completion, and -1 indicates failure.

Note

The function will convert the offset and the length align to cluster address.

Example

```
…
if((fd=open( (LPSTR)"A:\\test.bin", O_RDWR)) == -1)
        return -1;
DeletePartFile(fd, 0, 0x3000);//delete the data from 0 to 0x3000
close( fd);
…
```

2.42 InserPartFile

Insert a file into another file.

Function	Required Header
`InserPartFile`	vfs.h

```
int InserPartFile (int tagfd, int srcfd, unsigned long tagoff, unsigned long srclen)
```

Parameter

tagfd

The file want to insert data.

Srcfd

The file will be inserted.

`tagoff`

The target file address you want to insert data.

`srclen`

The source file data length you want to insert.

Return values

A return value of 0 indicates successful completion, and -1 indicates failure.

Note

The function will convert the offset and the length align to cluster address.

Example

```
...
if((tagfd=open( GLPSTR("A:\\tagfile.bin"), O_RDWR)) == -1)
        return -1;
if((srcfd=open( GLPSTR("A:\\srcfile.bin"), O_RDWR)) == -1)
        return -1;
offset = 0x800;
length = 0x5000;
InserPartFile(tagfd, srcfd, offset, length);
close(tagfd);
close(srcfd);

unlink(GLPSTR("A: \\srcfile.bin "));
```

3 Function Requirements

We need some functions for getting system time. We list the spec here:

3.1 UserGetDate

Function Interface

```
void UserGetdate (struct dosdate * dd);
```

Structure Description

```
struct dosdate    {
     unsigned short year;
     unsigned char monthday, month;
};
```

Function Definition Demo

```
void UserGetDate (struct dosdate *dd)   {
     // function body should be adapted by user
     dd->year = 2004;
     dd->month = 8;
     dd->monthday = 23;
}
```

3.2 UserGetTime

Function Interface

```
void UserGetTime (struct dostime * dt);
```

Structure Description

```
struct dostime    {
  unsigned char minute, hour, hundredth, second;
};
```

Function Definition Demo

```
void UserGetTime (struct dostime *dt)   {
     // function body should be adapted by user
     dt->hour = 16;
     dt->minute = 54;
     dt->second = 37;
     dt->hundredth = 0;
}
```

4 Limits and Suggestions

Some limits exist in this embedded file system. We list them as following.

4.1 Maximum Open File Number

There are only three file node structure variables in this file system. So at the same time only three files can be opened at most.

4.2 Implied Open Operations

Some functions impliedly require file node structure variables to work properly. With out enough file node structure they will report error. These requirements for file node structure variables are listed in the following table("-" means file node is visibly required):

Function name	Require file node structure variables number
open	1
close	-
read	-
write	-
lseek	-
rmdir	1
mkdir	1
chdir	1
getcwd	0
unlink	-
rename	2
stat	1
fs_init	0
_getfserrcode	0
_clserrcode	0
_findfirst	1
_findnext	1
_copy	2
_format	0
_setfattr	1
_deleteall	1
_devicemount	0
_deviceunmount	0
_deviceinfoget	0
_getdiskfree	0

4.3 Copy Operation Performance

We can not make sure which part of the SRAM is available to work as a data buffer when user is calling *_copy()* to duplicate a file. So we declared a 64 byte length unsigned char array as buffer in *_copy()* function (The array size is limited by the size of the memory size on board). It means than we can not make a high performance when user is trying to duplicate a file by *_copy()*. We strongly suggest the user to write a new copy function instead of the *_copy()* in your application.

4.4 Maximum Pathname String Length

The maximum pathname string length is 255 bytes, includes the drive letter and the ':' character and the backslashes but not includes the terminal zero character.